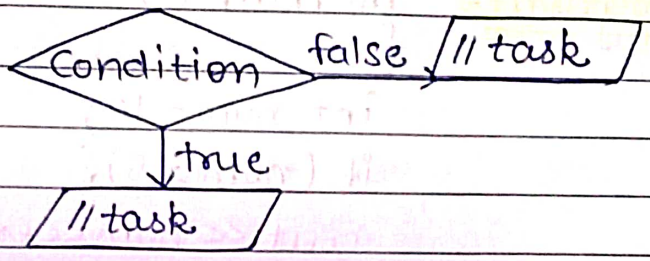
28/08/2023
Monday

* CONDITIONAL OR DECISION MAKING STATEMENTS -

Conditional statements are used to check whether the given condition is true, do the given task and if the condition is false, execute the another given task.



Decision-making statements -

1. if statement
2. if-else statement
3. Nested if statement
4. if-else-if statement
5. Switch statement
6. Conditional or ternary operator
7. Jump statements →
   - break
   - continue
   - goto
   - return

1. **if statement →**

Syntax →

```
if (condition)
{
    // logic
}
```

Program →

```
int main()
{
 int num = 10;
 if (num > 5){
  cout << num <<endl;
   }
}
```

Output = 10

2. **if - else statement →**

syntax →

```
if (condition)
{
  // logic
}
else
{
  // logic
}
```

Program -

```
int main()
{
  int age = 19;

  if (age >= 15.5 && age <= 18.5)
  {
    cout << "eligible for NDA" << endl;
  }
  else
  {
    cout << "Go for CDS" << endl;
  }
}
```

Output = Go for CDS.

3. if - else - if statement →

Syntax →

```
if (condition)
{
  // logic
}
else if
{
  // logic
}
else if
{
  // logic
}
else{
  // logic
}
```

Program -

```
int main( )
{
 int age=21;

if (age>=15.5 && age<=18.5)
 {
   cout<<"eligible forNDA"<<endl;
 }
else if (age>=20 && age<=25)
 {
cout<<"eligible for CDS & AFCAT"<<endl;
 }
 else
 {
   cout<<" not eligible"<<endl;
 }
}
```

Output= eligible for CDS & AFCAT

4. nested-if statement →

Syntax →

```
if (condition 1){
      //logic
     if (condition2){
       // logic
     }
}
else
 {
   // logic
 }
```

Program –

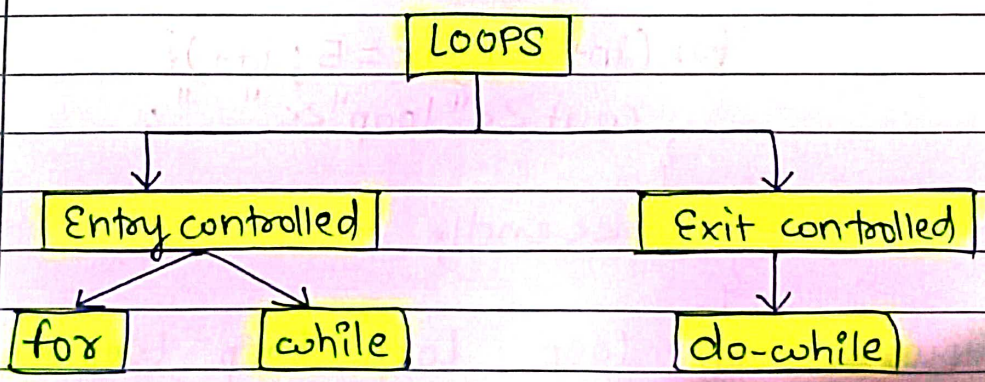
```
int main()
{
  int a=10, b=5, c=2;

  if (a<5){
      cout<<endl;
      if (c==2){
      cout<< c <<endl;
      }
    }
  else
    {
     cout<<"namaste"<<endl;
    }
```

Output = namaste

* **LOOPS →**

There is need to perform some operation more than once. Then, loops come into use when we need to repeatedly execute a block of statements.

For instance, we want to print "namaste duniya" 100 times, we will use loop.



__/__/____

- **Entry-controlled loops -**

The test condition is tested before entering the loop body. for loop and while loop are entry controlled loops.

- **Exit-controlled loops -**

The test condition is tested at the end of the loop body. Therefore, the loop body will execute at least once irrespective of whether the condition is true or false. The do-while loop is exit-controlled loop.

→ **FOR LOOP →**

Syntax →

```
for (initialization; condition; updation)
 {
    // loop body
 }
```

Program →

```
int main(){
    for(int i=1; i<=5; i++){
        cout<<"loop"<<" ";
    }
    cout<<endl;
}
```

Output = loop loop loop loop loop

→ NESTED FOR LOOP -

Syntax -
```
for (initialization; condition; updation)
{
    // logic
  for (initialization; condition; updation)
  {
     // logic
  }
}
```

Program -
```
int main() {

for (int i=0; i<2; i++) {
cout << endl << "outer loop" << i << endl << endl;

for (int J=0; J<2; J++) {
cout << " inner loop " << J << endl;
}
}
```

Output -
```
Outer loop 0

inner loop 0
inner loop 1

Outer loop 1

inner loop 0
inner loop 1
```

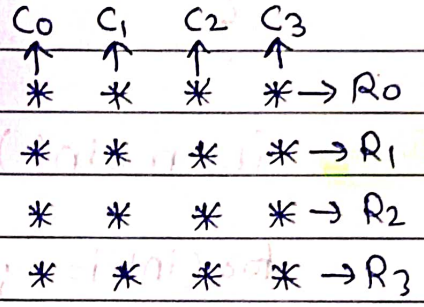
* **PATTERN PRINTING →**

Step 1 - Count the number of rows (outer loop)

Step 2 - See what is happening in each row (inner loop)

For instance,

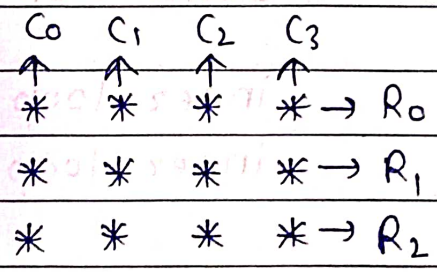
1. **Square Pattern →**



No. of rows = 4

$R_0 \rightarrow$ 4 star
$R_1 \rightarrow$ 4 star
$R_2 \rightarrow$ 4 star
$R_3 \rightarrow$ 4 star

2. **Rectangle pattern -**



No. of rows = 3

$R_0 \rightarrow$ 5 star
$R_1 \rightarrow$ 5 star
$R_3 \rightarrow$ 5 star

3. **Hollow square –**

C0 C1 C2 C3 C4

* * * * * → R0
* * → R1
* * → R2
* * → R3
* * * * * → R4

No. of rows = 5

R0 → 5 star
R1 → 1star, 3space, 1star
R2 → 1star, 3space, 1star
R3 → 1star, 3space, 1star
R4 → 5 star

4. **Half pyramid –**

R0 *
R1 * *
R2 * * *
R3 * * * *
R4 * * * * *

No. of rows = 5

R0 → 1 *
R1 → 2 *
R2 → 3 *
R3 → 4 *
R4 → 5 *

Formula = row no. + 1

5. **Invested half pyramid** –

R0 → * * * * *
R1 → * * * *
R2 → * * *
R3 → * *
R4 → *

Total rows = (n = 5)

R0 → 5star
R1 → 4star formula = n − row no.
R2 → 3star
R3 → 2star
R4 → 1star

6. **Numeric half pyramid** –

R0 → 1
R1 → 1 2
R2 → 1 2 3
R3 → 1 2 3 4
R4 → 1 2 3 4 5
↓ ↓ ↓ ↓ ↓
C0 C1 C2 C3 C4

Total rows = (n=5)

R0 → 1 (1 char) Formula = row no. + 1
R1 → 1 2 (2 char) To print = column no. + 1
R2 → 1 2 3 (3 char)
R3 → 1 2 3 4 (4 char)
R4 → 1 2 3 4 5 (5 char)

7. **Inverted numeric half pyramid –**

	C_0	C_1	C_2	C_3	C_4
	↑	↑	↑	↑	↑
R_0 →	1	2	3	4	5
R_1 →	1	2	3	4	
R_2 →	1	2	3		
R_3 →	1	2			
R_4 →	1				

Total rows = (n=5)

R_0 →	1	2	3	4	5	(5char)
R_1 →	1	2	3	4		(4char)
R_2 →	1	2	3			(3char)
R_3 →	1	2				(2char)
R_4 →	1					(1char)

Formula = n – row no.

To print = column no. + 1